Class: CS436 Fall2017

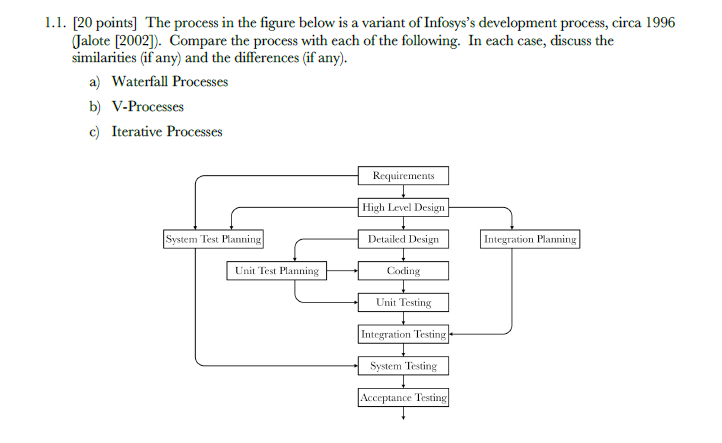
Assignment: HW1

Due time: Sep 7, 2017 11:30 PM

Student: Dong Liang

Professor : Ravi Sethi

TA/Graders : Jacob Combs



1. **Waterfall**

Similarities: Both waterfall and this process in the figure are sequential activities (step by step). For instance, in the figure, the high-level design is after the clarify of requirements and the detailed design is after high level design and the coding is next and so on.

Differences: For the pure waterfall, it does not have customer involvement after requirements and requirements changed before the project completed. However, the process in the figure does have feedback from customer at very first and can test it multiple times during the steps to avoid a crazy entire change at the end of the process.

1. **V-Process**

Similarities: Both V Process and this one in the figure use the tests to support the development (developer work based on the test).

Differences: For the V-Process, test and development progress in parallel. However, the process in the figure doesn’t have this feature.

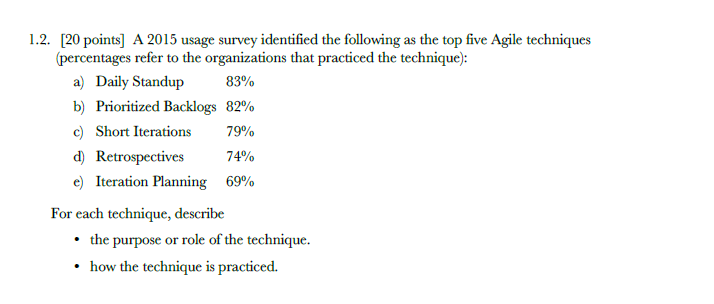
Also, the V Process was not designed in a linear axis. But the main process of the one in the figure (from “Requirements” to “Acceptance Testing”), is linear.

1. **Interactive Process**

Similarities: Both emphasize what to build or the customer’s requirements.

The proof is the process in the figure starts with the “Requirements”.

Differences: For interactive process, the customer’s feedback and corrections happen in each iteration while this process doesn’t have such a thing.



1. Daily Standup

The purpose is to self-organize as a team.

How to practice it: Meet daily at the same time with the team. The time for each meeting should be at least 15 minutes.

1. Prioritized Backlogs

The purpose: It is used to keep track of the prioritized work have been done. This gives the develop team the requirements in detail in each step.

How to practice it: Continuedly elaborate the details of requirements.

1. Short Iterations

The purpose: The short iterations gives the team a better understanding of how much work can be done in a period. And it turns out to be high quality code.

How to practice it: The length of an iteration should be discussed in the meetings. And people could minimize the size of user stories to schedule a short iteration.

1. Retrospectives

The purpose: Always happen at the end of one iteration and it gives the team some idea to make them more efficient in the next iteration (To be more efficient for the work in the future).

How to do it: Have a meeting at the end of one iteration. Teammates should talk about the potential improvement they could make. And if the suggestion or plan is agreed then take the improvement for the next iteration.

1. Iteration Planning

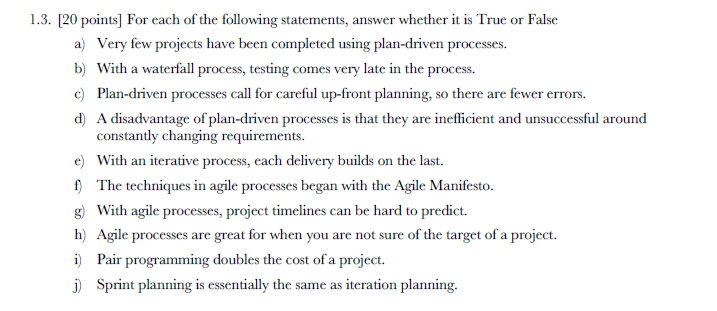
Roles: A team should determine how the work should be done for the next iteration.

This must be decided and planed by all members of the team.

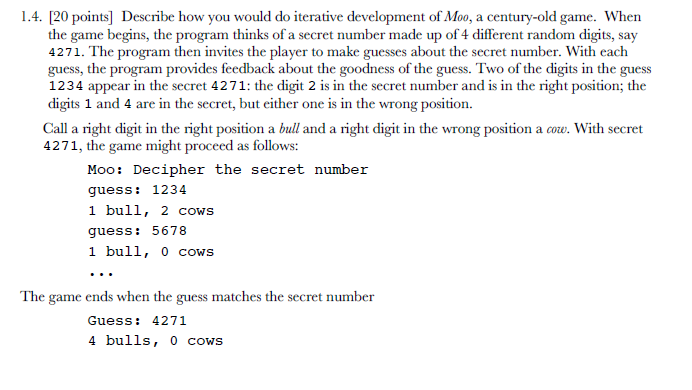
How to do it: Members of a team planed and discuss together before an upcoming iteration. When multiple choices happen (one task can be done different ways), team members should use the user stories and their own experience to make the choice.

Note: My answering of question 1.2 is after the reading of this article.

Linke : https://learn.techbeacon.com/units/5-most-common-agile-techniques



1. False
2. True
3. True
4. True
5. False
6. True
7. False
8. True
9. False
10. False



1. Ask the customer what is the requirements.

In this case, the customer should be the player of the game. And the requirement could be the type of this game (A GUI that can run on windows/MAC? Or a game on website?)

1. Define the product.

Be clear what kind of a game that I am going to build. Ask the player about more detail of the game. For example, do the players want a rank system?

1. Based on the requirements, write down the spec of the project.

My spec should have solutions to the player’s need right now. For example, if they just want a GUI that can run on windows/mac, then maybe I would pick java/python to do it.

1. Check if my design is what the players want.

I could show the players a demo. And tell them my ideas of the game. Then maybe they will give me more details or not. After the check with customers I could start writing the code of the game.

Since the product is a game, so the test would be playing around. And I could always present a demo to the customers. And if they are not happy with some details, then go back and modify the spec and change my code until the product satisfy my customers.